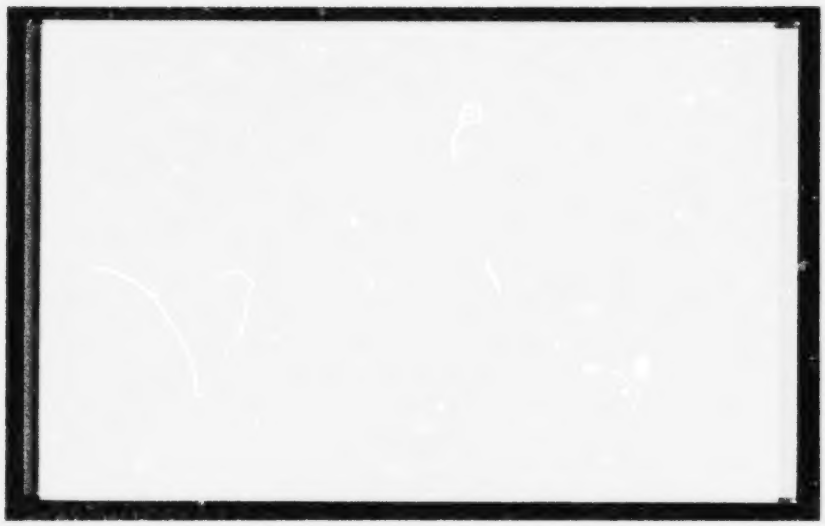
UNIVERSITÉ DE MONTRÉAL



CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

THE MULTIPLE VEHICLES MANY TO MANY ROUTING PROBLEM WITH TIME WINDOWS

by

Jacques Desrosiers*, Yvan Dumas**
and François Soumis***

* Département de l'enseignement des méthodes quantitatives, École des Hautes Etudes Commerciales, Montréal, Canada H3T 1V6.

** Centre de recherche sur les transports, Université de Montréal, Montréal, Canada H3C 3J7.

*** Centre de recherche sur les transports, Université de Montréal, Montréal, Canada H3C 3J7 and Ecole Polytechnique de Montréal, Canada H3C 3A7.

Université de Montréal

Centre de recherche sur les transports - Publication #362
École des Hautes Etudes Commerciales - Cahier du GERAD #84-13

Décembre 1984

Dépôt légal - 3e trimestre 1985

ABSTRACT

The multiple vehicle many-to-many routing problem is solved by mini-clustering first and optimal routing second. A heuristic mini-cluster algorithm groups together customers travelling at the same time in the same vehicle. The routing problem is optimally solved using a column generation algorithm. Problems with up to 200 customers and 85 mini-clusters are easily solved. Larger problems can be solved by dividing the day into time slices and applying the algorithm several times.

1. INTRODUCTION

This article examines the optimization of vehicle routes is a large scale multi-vehicle many-to-many system. The methodology is presented in the context of the transportation of the handicapped, but may be adapted to goods transportation by truck.

1.1 Problem description

For a given day, we have a set of transportation requests, a homogeneous vehicle fleet and pieces of work for drivers. Each transportation request involves 1) a pick-up stop and a delivery stop 2) a given number of clients to be transported, 3) a departure time interval and an arrival time interval 4) pick-up and delivery times. Vehicles have capacity limits on the number of wheelchairs, and an independent capacity for ambulatory passengers. The drivers pieces of work are known in advance. Each piece of work involves a depot with departure and arrival times defining the work period. We consider the case with several depots.

Distances and times between the various stops are given. The problem consists of minimizing the number of pieces of work used and then for this number of pieces of work, minimizing the total distance travelled while satisfying the following constraints:

1) visiting constraints:
   Each pick-up and delivery stop is visited only once.

2) time window constraints:
   The time constraints must be satisfied at each stop.

3) capacity constraints:
   Vehicle capacity cannot be exceeded.

4) depot constraints:
After a working period, each vehicle returns to the depot from which it departed.

5) pairing constraints:
For a given client, the same vehicle must visit the pick-up and delivery points.

6) precedence constraints:
Each client must be picked up before being delivery.

7) availability constraints:
The availability of driver pieces of work must be respected.

1.2 Literature review

Bodin et al [2] in their state of the art article describe three algorithms for the multi-vehicle problem. The NEIGUT/NBS procedure shedules one vehicle at a time and determines the best customer to insert into the route. When no more customers can be added to a route, this route is frozen and the process is repeated over the remaining set of unassigned customers. This is a sequential insertion procedure and generally, the routes generated first are the most productive, while the routes generated last tend to be inferior in quality.

The MIT procedure [7] is a concurrent procedure. The day is split a priori into time periods of 20-30 minutes in length. Assuming that partial vehicle schedules exist over all proceeding periods, the MIT procedure forms clusters for the customers for each of the vehicles in the given period. This is accomplished via a special spanning tree algorithm.

Bodin/Sexton [3] procedure is a clustering/concurrent insertion procedure. It begins by finding small subsets of customers who look as though right be serviced by the same vehicle (about the same desired delivery times, close delivery locations and pickup locations). These subsets of customers serve as seeds for the formation of vehicle routes. Other customers are attached to these clusters via the iterative use of a transportation problem. A 1-opt algorithm is also utilized to move customers between routes in order to reduce total customer inconvenience.

Another procedure has been developped at the university of Montreal. The work of Roy et al [10] is also based on a concurrent insertion procedure. This algorithm takes a set of known requests and simultaneously constructs routes for all vehicles starting at the beginning of the day by using proximity measures. It can also insert new requests into a set of existing routes.

1.3 Proposed methodology

In a methodology of the "cluster first-routing second" type a cluster is formed from the set of customers assigned to a vehicle and routing is carried out separately for each vehicle. The routing problems are usually fairly easy to solve. However, the most important decisions are taken at the clustering stage and it is very difficult to globally construct a set of good clusters.

We adopt thus "cluster-first-routing second" approach while moving a part of the clustering problem into the routing problem. In the first step, mini-clusters are formed from groups of clients travelling at the same time in the same vehicle. At the second step, routes for all the vehicles are constructed simultaneously, dividing the mini-clusters between vehicles. We propose an application of thus approach in which the mini-clusters are constructed starting from a feasible solution:

1A. The first step uses the single vehicle algorithm (Desrosiers, Dumas, Soumis [4]) to optimize each of the routes in the initial set provided.

1B. These routes are divided into mini-clusters of customers.

2A. In the second step, integer linear programming with column generation (Desrosiers, Soumis, Desrochers [6]) is used to simultaneously reallocate mini-clusters between vehicles so as to minimize the number of pieces of work and travel costs. Decomposition on overlapping time periods is used for large problems.

2B. We finally open up the mini-clusters and reoptimize each route by proceeding as for step 1.

In conjunction with the Roy et al [10] heuristic, this procedure has been used to analysis the impact on vehicle routing of various operating rules for transportation system for the handicapped persons (Bélisle et al [1]).

With respect to the original problem to be solved, and as described in the following sections, the single vehicle algorithm deals with the time window, visiting, capacity, precedence and depot constraints; pairing and availabity constraints are automatically satisfied. The column generation algorithm handles the time window, visiting, availability and depot constraints. The contribution of the algorithm presented in this paper is to provide a very high quality solution to the multi-vehicle many-to-many routing problem with time windows. In addition, the column generation methodoly presented in [6] is extended. The algorithm developed handles problems with several depots and with availability constraints.

2. CONSTRUCTION OF MINI-CLUSTERS

Steps 1A and 2B of our procedure require the optimization of routes one by one. This is carried out using the single vehicle dial-a-ride algorithm described in Desrosiers, Dumas, Soumis [4].

A route usually serves from 5 to 20 clients i.e. the vehicle visits from 10 to 40 stops. The optimal route minimizing the distance travelled by a single vehicle visiting all stops is obtained by dynamic programming. It should be stressed that in the problem with a single vehicle, the pairing constraints are automatically satisfied and a piece of work is available (supplied by the feasible solution). A travelling salesman problem with additional time window, capacity and precedence constraints must therefore be solved. A time variable is associated with the time of visiting each stop: as these time variables are strictly increasing along the route formed, subtours are automatically eliminated.

Dynamic programming is not efficient for solving pure travelling salesman problems of this size. The efficiency of our method is due to the use of criteria for the elimination of states which do not satisfy the additional constraints. At a given iteration, a state (S,i) is defined where S is the unordered set of stops visited by the vehicle and where $i \in S$ is the last stop visited. At the following iteration, a new stop j is added. it is then easy to eliminate a state ($S \cup \{j\}$, j) which does not satisfy the time window, precedence or capacity constraints. In addition, as the vehicle has to visit every stop, state elimination can be improved by considering two so far unvisited nodes, if the two nodes with the earliest time windows cannot be visited after node j so as to satisfy the time constraints, then state ($S \cup \{j\}$, j) is eliminated, thus reducing the combinational development of the algorithm. (With 3 stops, there are 6 permutations to be verified, while with 2 stops there are only 2). Experience has shown that execution time is a linear function of the number of stops.

After optimizing the total distance travelled on a route, this is divided into mini-clusters each time the vehicle becomes empty. A mini-cluster groups together on average less than 4 requests for clients travelling at the same time in the same vehicle. Within a mini-cluster, the time window, precedence, pairing and capacity constraints are all satisfied.

To construct the mini-clusters, other objective functions could be used with the single vehicle algorithm. For example, the cost of movements with clients in the vehicle could be minimized. A different set of mini-clusters would be obtained.

A time window for the starting time is then associated with each mini-cluster based on the time windows of the stops within the mini-cluster. The problem to be solved at step 2A becomes a routing problem with time window constraints on the mini-clusters and availability constraints on the pieces of work.

3. ROUTING AND SCHEDULING ON MINI-CLUSTERS

The problem in step 2A involves mini-clusters $i \in P$ each with a duration and a cost (time and distance between the first and last stop in the mini-cluster) as well as a time window $[a_i, b_i]$ on the visit starting time. We also have for the drivers pieces of work $k \in K$ with departure points s_k and arrival points e_k as well as a work period defined by $[a^k, b^k]$.

Pieces of work are given a priori because collective agreements usually require driver schedules to be set several days in advance. A driver may be assigned to several pieces of work, for example, a piece of work from 6.00 a.m. to 12. a.m. in the morning and another from 3.00 p.m. to 6.00 p.m. in the afternoon. We have Q_k drivers avelable for the piece of work of type k.

3.1 Formulation

The problem consists of finding vehicle schedules and routes which minimize the number of pieces of work and travel times In the formulation, we use a network structure (N; A) where the set of nodes is $N = \bigcup_k N^k$ with $N^k = P \cup \{s_k, e_k\}$ and the set of arcs is $A = \bigcup_k A^k$ with $A^k = I \cup (\{s_k\} \times P) \cup (P \times \{e_k\})$. An arc $(i, j) \in I \subset P \times P$ is an unproductive (i.e. empty) run carried out by a vehicle as it goes from the end of mini-cluster i to the beginning of mini-cluster j. Its duration t_{ij} and its cost (distance) c_{ij} may include respectively the duration and cost of mini-cluster i. An arc $(i, j) \in A$ is defined if the schedule permits it (i.e. if $a_i + t_{ij} < b_j$). Finally, a large fixed cost C is added to the cost of the arcs of the subsets $\{s_k\} \times P$, for all $k \in K$, to ensure that the number of pieces of work is minimized.

Two types of variables are used in the formulation: flow variables X^k_{ij} which are binary, $X^k_{ij} = 1$ indicate whether vehicle k (associated with the piece of work k) travels from node i to mode j, continuous time variables t_i, $i \in P$ associated with the visit starting time for the mini-clusters. Optimal routes respecting the scheduling and availability constraints are given by the solution of the following integer programming problem:

$$\text{Min} \quad \sum_{k \in K} \sum_{(i,j) \in A^k} c^k_{ij} \, X^k_{ij} \qquad (1)$$

$$\text{subject to:} \quad \sum_{k \in K} \sum_{j \mid (i,j) \in A^k} X^k_{ij} = 1 \qquad i \in P \qquad (2)$$

$$\sum_{j|(\ell,j)\in A^k} x^k_{\ell j} = \sum_{i|(i,\ell)\in A^k} x^k_{i\ell} \qquad \ell \in P,\ k \in K \tag{3}$$

$$\sum_{j\in N^k} x^k_{s_k j} = \sum_{j\in N^k} x^k_{j\ell_K} \qquad k \in K \tag{4}$$

$$\sum_{j\in N^k} x^k_{s_k j} \leq Q_k \qquad k \in K \tag{5}$$

$$a_i \leq t_i \leq b_i \qquad i \in P \tag{6}$$

$$x^k_{ij} > 0 \rightarrow t_i + t_{ij} \leq t_j \qquad (i,j)\in A^k,\ k \in K \tag{7}$$

$$x^k_{s_k j} > 0 \rightarrow a^k + t_{s_k j} \leq t_j \qquad j \in N^k,\ k \in K \tag{8}$$

$$x^k_{ie_k} > 0 \rightarrow t_i + t_{ie_k} \leq b^k \qquad i \in N^k,\ k \in K \tag{9}$$

$$x^k_{ij} \qquad \text{binary} \qquad i,\ j \in N,\ k \in K \tag{10}$$

In the objective function, costs are defined for all k K as $C^k_{ij} = c_{ij}$, $(i,j)\in I \subset P \times P$, $c^k_{s_k j} = C + c_{s_k j}$, $j \in P$, and $c^k_{ie_k} = c_{ie_k}$. Constraints (2)-(3) ensure that one and the same vehicle enters and leaves each stop. Constraints (4) are the depot constraints. Inequalities (5) model the availability limitations on the number of pieces of work. Constraint (6) establish the time intervals on the visit starting time of mini-clusters while constraints (7)-(9) describe the compatibility requirements between the routes and schedules.

Rather than solving the problem in terms of flow variables in the network (N; A), we formulate and solve it in terms of binary variables on the use of feasible paths in graphs (N^k; A^k). As it is impossible to enumerate every path, we solve the problem by column generation (Dantzig-Wolfe decomposition). The columns are generated by solving shortest path

problems with scheduling constraints on the networks $(N^k; A^k)$. The algorithm used is an extension of the method presented in Desrosiers, Soumis and Desrochers [6]. It can solve problems with several depots and availability constraints.

3.2 Column generation scheme

The formulation in terms of feasible routes is a partitioning problem with availability constraints on the pieces of work. It can be written:

$$\text{Min} \quad \sum_{k \in K} \sum_{r \in \Omega_k} C_r^k \, Y_r^k$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{r \in \Omega_k} \delta_{ri}^k \, Y_r^k = 1 \qquad i \in P \qquad (11)$$

$$\sum_{r \in \Omega_k} T_r^k \leq 0 \qquad k \in K \qquad (12)$$

$$Y_r^k \quad \text{binary} \qquad r \in \Omega_k, \; k \in K \qquad (13)$$

where

Ω_k : The set of routes satisfying the scheduling constraints of piece of work k.

Y_r^k : Binary variable, if route r of piece of work k is used, 0 otherwise.

C_r^k : Cost of route r of piece of work k (the sum of costs of arcs making up the route).

δ_{ri}^k : Binary constant, if route r of piece of work k goes through the mini-clusters i, 0 otherwise.

To solve the linear relaxation of this problem, rather than enumerating in advance all the sets Ω_k, column generation is carried out. In the terminology of Dantzig-Wolfe decomposition, the linear problem with a subset of variables will be called the master problem.

For a piece of work $k \in K$, a column of minimum marginal cost is generated by solving the shortest path subproblem with time window constraints on the network $(N^k; A^k)$. This subproblem is defined using the dual variables σ_i, $i \in P$ and Π_k, $k \in K$ of the master problem:

$$\text{Min} \quad \sum_{(i,j) \in A^k} c_{ij}^k \, x_{ij}^k$$

$$\text{subject to} \quad \sum_{j \in N^k} x_{\ell j}^k - \sum_{i \in N^k} x_{i \ell}^k = \begin{cases} +1 & \ell = s_k \\ 0 & \ell \in P \\ -1 & \ell = e_k \end{cases} \qquad (14)$$

$$a_i \leq t_i \leq b_i \qquad i \in P$$

$$x_{ij}^k > 0 \rightarrow t_i + t_{ij} \leq t_j \qquad i,j \in N^k \qquad (15)$$

$$x_{s_k j}^k > 0 \rightarrow a^k + t_{s_k j} \leq t_j \qquad j \in N^k \qquad (16)$$

$$x_{i e_k}^k > 0 \rightarrow t_i + t_{i e_k} \leq b^k \qquad i \in N^k, \; k \in K \qquad (17)$$

$$x_{ij}^k \text{ binary}$$

where

$$c_{ij}^k = \begin{cases} c_{ij}^k - \Pi_k & i = s_k \\ c_{ij}^k - \sigma_i & i \in P \end{cases} \qquad (18)$$

3.3 Solution method

The optimal integer solution of the subproblems is obtained by dynamic programming. This method is particularly efficient in this case because each vehicle visits only a few mini-clusters (nodes). We use the algorithm described in Desrosiers, Pelletier, Soumis [5].

The master problem is solved using the simplex algorithm. This method gives the dual variables σ_i, $i \in P$ and Π_k, $k \in K$ necessary for column generation and lends itself well to reoptimization each time new columns are generation. The integer solution of the master problem is described in section 3.4.

In this application, several strategies were used to reduce execution time. Because each sub-route of a feasible route is also feasible at a lesser cost, the equations of the form (=) in the master problem could be transformed into inequalities (≥). Better numerical stability was thus obtained.

At the beginning of the algorithm, an initial solution is available, given by the feasible routes of step 1A. An initial basis is obtained by adding null variables using step 1 of the simplex.

The solution of the master problem by simplex often produces integer or partially integer solutions. But, a typical basis contains 60% to 80% null variables, which unfortunately causes a large number of degenerate simplex iterations for each new column. To increase the number of non-null variables in the basis, a negative perturbation was applied to the covering constraints and a positive perturbation to the availability constraints. The solution time for the perturbed problem is much shorter, and the gap between the solutions of the perturbed problem and the original linear problem is small. When the perturbations are removed, the time required to find the solution to the original problem is small. Finally at each call of the simplex, the columns whose reduced cost is too high are eliminated.

For a set of multipliers, $\sigma_i, i \in P$ and Π_k, $k \in K$, a strategy which generates several columns was adopted. For a given subproblem, at most a number of columns equal to the availability Q_k of this piece of work is generated. Columns are generated one by one by withdrawing from this subproblem and the following ones, mini-clusters already covered, after each column has been generated.

The first subproblem to be handled after each simplex call is chosen cyclically. If thus first subproblem does not generate a column of negative marginal cost, it is temporarely excluded from the generation step. It is reinstated later to ensure the optimality of the solution of the master problem.

3.4 Obtaining integer solutions

The sum of flows on the exist arcs from all s_k, $k \in K$ towards all mini-clusters $i \in P$ gives the number of pieces of work used. In our application, the fixed cost C is sufficiently large so that it is never worthwhich to increase the number of pieces of work in order to reduce travel costs. When the total number of pieces of work $\sum_{k \in K} \sum_{r \in \Omega_k} y_r^k = pw$ is non-integer in the optimal linear solution of the master problem, a cut is added to round this number up to at least the next integer:

$$\sum_{k \in K} \sum_{r \in \Omega_k} y_r^k \geq \lceil pw \rceil$$

The master problem is then reoptimized generating new columns as needed. For each subproblem, the objective function is modified by the new dual variable α associated with this cut:

$$\text{Min} - \alpha + \sum_{(i,j) \in A} {}^k c_{ij}^k \, x_{ij}^k$$

If the solution is still non-integer, we use a branch-and-bound tree. (The second cut on the travel cost described in [6] is not used here because it is not efficient in this context). Two types of branching may be used at a node in the tree. A branching related to the use of pieces of work is the first priority. If $\sum_{r\in\Omega_k} Y_r^k = q_k$ is fractional, two branches are created in the tree, with on one branch $\sum_{r\in\Omega_k} Y_r^k \leq \lfloor q_k \rfloor$ and $\sum_{r\in\Omega_k} Y_r^k \geq \lceil q_k \rceil$ on the other. The branch containing $\lfloor q_k \rfloor$ is treated first if the fractional part is less than 0.4. The master problem is reoptimized generating new columns as required, the dual variable associated with this constraint is transferred to the subproblem k in a similar way as the cut on the total number of pieces of work. The piece of work chosen for branching is the one whose fractional part is closest to 0.25 or 0.75 (as those with fractional parts which are too low or too high make little difference to the integer solution).

After branching on pieces of work, if the solution is still non-integer, branching is carried out on the binary variables Y_r^k. This is not done by fixing a Y_r^k variable at 0 or 1, but by setting at 0 or 1 the variables X_{ij}^k associated with the arcs which make up the route. Suppose that the variables

$$X_{s_k 1}^k = X_{12}^k = X_{23}^k = X_{3e_k}^k$$

an associated with the variable $0 < Y_{r_k}^k < 1$. We create 5 branches which are successively defined as ($X_{s_k 1}^k = X_{12}^k = X_{23}^k = X_{3e_k}^k = 1$), ($X_{s_k 1}^k = X_{12}^k = X_{23}^k = 1$, $X_{3e_k}^k = 0$), ($X_{s_k 1}^k = X_{12}^k = 1$, $X_{23}^k = 0$), ($X_{s_k 1}^k = 1$, $X_{12}^k = 0$) and ($X_{s_k 1}^k = 0$).

With this method we can transfer information from the branching to the network of the subproblem k (see [6]). The fractional variable Y_r^k chosen is the one whose components most often appear in the current basis. Exploring first the branch for which all flow variables are set at 1, a good quality integer solution is found rapidly. Exploration of the tree terminates when the number of pieces of work is optimal and when the gap with respect to optimal travel time is less than 1%.

3.5 Decomposition on overlapping time slices for large problems

When the routing problem on mini-clusters becomes too large, the day is divided into successive time slices or periods and a sequence of smaller problems is solved. The division of the day into time slices is carried out according to two criteria: an overlap time interval of 20 to 30 minutes between successive slices, and a number of mini-clusters varying between 30 and 40 per slice (including the mini-clusters in the overlap time). The reduced size of the problems solved for each period permits rapid handling by the column generation algorithm.

At the beginning of the algorithm we have a global feasible solution. For each route in this solution, we have nodes of the network before, during and after the being considered period. For a given route, the associated piece of work is replaced by a piece of work of reduced length. The starting place and time for the piece of work are defined as the last stop and time which the driver completed prior to the beginning of the time slice. Similarly, the finishing place and time are defined as the first stop and time to be visited after the end of the time slice. Pieces of work which were previously identical, are now different; we have a subproblem for each route of the feasible solution which covers the mini-clusters in this period. This procedure retains after each time slice a new feasible solution which is used to define the problem for the following time slice.

For a given time slice, if a vehicle comes from the depot, a fixed cost is assigned to the corresponding arc; if the vehicle is merely carrying out the continuation of a piece of work, there is no fixed cost. To take account of this, the branching procedure on fractional variables was modified slightly to favour the use of columns with zero fixed cost.

4. TESTS

In the following sections, we present the numerical results for 6 problems used to test the proposed methodology.

4.1 Problem description

The six test problems come from the canadian cities of Sherbrooke (SHE), Totonto (TOR) and Montreal (MTL). The alphanumeric code identifying each problem indicates the city and the number of transportation requests. Table 1 describes the main characteristics of these six problems.

The number of requests varies from 86 to 190 for the four smallest problems and there are 410 and 880 requests for the two large Totonto problems. For each problem, the number of stops to be visited by all the vehicles is roughly twice the number of requests. This is because although there are certain common stops for the departure and arrival points of clients, additional nodes for the depots associated with pieces of work have to be added. In the routing problem on mini-clusters, the number of nodes to be visited is reduced by a factor of from 2.2 to 3.5. Although the average number of requests making up a mini-cluster is less than 4, the distribution of the size of mini-clusters shows that there can be over 10 requests in some mini-clusters. The problem TOR185 handles morning requests only, while the other problems handle requests for a full day. Problem TOR410 includes regular requests in this city. Occasional requests are then added to form problem TOR880. These two problems amongst others were used for the evaluation of various transportation system scenarios for the handicapped by Bélisle et al [1].

Table 1: Problem description

IDENTIFICATION (a)	SHE86	TOR185	MTL158	MTL190	TOR410	TOR880
Number of mini-clusters	36	56	66	85	115	282
Size distribution of mini-clusters						
(1-4)	30	40	59	77	81	212
(5-10)	6	14	7	8	31	66
(11-14)	0	2	0	0	3	4
(average)	2.4	3.3	2.4	2.2	3.5	3.1
Visit starting time distribution of mini-clusters						
(5-10h)	11	56	27	29	54	105
(10-14h)	9	0	1	1	0	54
(14-18h)	13	0	37	43	61	93
(18-24h)	4	0	1	12	0	30
Network size (N; A) (b)						
(nodes)	56	70	90	119	153	310
(arcs)	672	906	1835	3145	4979	(c)
Time windows on mini-clusters						
(average)(d)	19	22	15	16	21	21
(max.)	40	40	30	30	45	45
Duration of mini-clusters						
(average)(e)	33	60	42	40	65	60
(max.)	113	192	128	128	195	235
Duration of arcs						
(average)(d)	39	66	57	58	80	83
(max.)	126	226	182	182	243	291
Different pieces of work	10	7	12	17	19	19
Total availability	12	28	24	34	69	60

(a) code identifying the city and number of requests
(b) the Toronto problems contain two different depots
(c) for this very large problem, solved by decomposition, the number of arc is estimated to be over 30,000
(d) with a minimum of 0 minutes
(e) with a minimum of 5 minutes

Although the average length of the arcs plus the duration of mini-clusters is from 6 to 8 times greater than the width of the time windows, the networks (N; A) are not acyclical. For example, mini-clusters with a single client have time windows of from 30 to 45 minutes, but they can be of very short duration (5-10 minutes) and the network includes arcs of from 0 to 5 minutes.

Finally, the number of different pieces of work gives the number of subproblems generating feasible routes when mini-cluster routing is carried out with a single problem (7 to 17 for the first 4 problems). When decomposition into time slices is carried out, the number of subproblems per slice is roughly equal to the minimum number of pieces of work used in the initial feasible solution (each defining a subproblem) plus the number of different unused pieces of work: this number is greater than 40 for problems TOR410 and TOR880

4.2 Test results

The first four problems were solved without decomposition into time slices at the mini-cluster routing step. Table 2 summarizes the results of the various steps in the procedure.

The value of the objective function is given in two parts: the number of pieces of work, and the total travel time in minutes (a function of distance). In relation to the initial solution, the average reduction in travel time is given as follows: 3% after individual route optimization, 10% after mini-cluster routing, and 11% after the final optimization of each route. For problems MTL158 and MTL190, the column generation algorithm also produced reductions of 4% and 9% respectively in the number of pieces of work used.

Table 2: Results of problems solved without decomposition at the mini-cluster routing stage

IDENTIFICATION (a)	SHE86		TOR185		MTL158		MTL190	
Initial solution (b)	10	1158	26	4312	24	3359	34	4323
Step 1A	10	1125	26	4101	24	3242	34	4202
Step 2A	10	1084	26	3686	23	3094	31	3880
Step 2B	10	1081	26	3614	23	3043	31	3801
Gain	0%	7%	0%	16%	4%	9%	9%	12%
Computation time in seconds on a CDC 173 computer								
Step 1A and 2B (c)	19	(20)	80	(52)	23	(46)	31	(62)
Step 2A								
Simplex (d)	8	(2)*	41	(6)	36	(1)*	104	(3)
Subproblems (e)	3	(131)	31	(281)	33	(414)	54	(536)
File management (f)	33		44		79		78	
Total	63		196		171		267	

(a) code identifying the city and number of requests
(b) objective function: number of pieces of work and travel time in minutes
(c) in brackets, the number of routes optimized
(d) time for simplex and number of nodes explored in the tree;
* for optimality
(e) time for the subproblems, and number of columns generated
(f) creation and update of files of routes, mini-clusters, pieces of work and time subtables

In the computation time section of the table, note the superior performance of the optimization algorithms. A fairly large proportion of the total execution time is due to file handling; further work is necessary to improve the efficiency of this aspect. Concerning the integer solution of the master problems at the column generation stage, the integer optimal solution was obtained for problems SHE86 and MTL158 respectively at the first and second nodes of the tree. For problems TOR185 and MTL190, the integer solutions presented are optimal for the number of pieces of work and 0.19% (6th node) and 0.10% (3rd node) from the continuous simplex solution for travel times. The number of columns generated by the subproblems is high in comparison with results previously published on the column generation methodology. This increased number of columns is due to the large number of subproblems to be solved and also to the elimination of simplex variables; certain columns must therefore be generated several times.

We have evaluated the effect of tree strategie used to reduce computing time and memory requirement. The pertrubation strategy reduces the computing time by 10%. With the variables elimination, only 40% of the variables are keepted in memory but computation time remain the same. When we enclude the initial solution strategy the computing time is reduce by 40%. This paper presents the results obtained using the tree-strategy approach. When the inititial solution is used the number of eliminated columns is lower but the elimination strategy computing time overhead remains important. So computing time can be reduce by 12% without elimination strategy, nevertheless the memory requirement for variables is increased by 38%.

A further optimization phase was tested by repeating our procedure with mini-clusters constructed from the solution of step 2B. Improvement was low or zero, i.e. less than 1% of travel time. The solutions produced are thus of very high quality and this post-optimization phase was not included in the procedure.

Table 3 shows the results of problems where mini-cluster routing was carried out by decomposition into overlapping time slices.

Table 3: Results of problems solved by decomposition at the mini-cluster routing stage

IDENTIFICATION (a)	TOR185	MTL158	MTL190	TOR410	TOR880
Initial solution (b)	26 4312	24 3359	34 4323	37 8791	53 19273
Solution without decomposition	26 3614	23 3043	31 3801	—	—
Number of time slices	2	2	3	4	13
Solution with decomposition	26 3659	23 3043	31 3875	31 7179	48 15765
Gain vs initial solution	9% 15%	4% 99%	9% 10%	16% 18%	9% 18%
Loss vs. solution without decomposition	0% 1%	0% 0%	0% 2%	—	—
Computation times in seconds on CDC 173 computer					
Steps 1A and 2B (c)	83(52)	23(46)	30(62)	23(62)	483(96)
Step 2A (d)	77	46	49	150	770
File management (e)	55	90	102	123	279
Total	215	159	181	504	1532

(a) code identifying the city and number of requests

(b) objective function: number of pieces of work and travel times in minutes

(c) in brackets: the number of routes optimized

(d) total time for simplex and subproblems for all decomposition periods

(e) creation and update of files

For the first 3 problems, i.e. TOR185, MTL158 and MTL190, comparisons can be made with solutions obtained without decomposition. For these three problems, the same number of pieces of work is obtained. Travel times are identical for problem MTL158, and are slightly higher (by 1% and 2% respectively) for problems TOR185 and MTL190. computation times for the column generation algorithm are lower for two out of three problems; however there is a slight increase in the time required for file management. Overall, solutions remain of about the same quality and the decomposition into time slices is an efficient alternative for the solution of large problems.

The two large problems TOR410 and TOR880 were both solved using this decomposition approach. Comparison with the initial solutions given by Roy et al [10], shows that the number of pieces of work can be significantly reduced (by 16% and 9%), as can the total travel time (by 18% and 18%).

Computation times for the optimization of routes one at a time by dynamic programming as described in [4] are fairly high, i.e. nearly 5 seconds on average. The 31 routes of problem TOR410 contain on average 13 requests while the 48 routes of problem TOR880 contain only 18 on average. The difficulty of these problems is due to the proximity of several origins and/or destinations for requests. A study will be undertaken to limit the number of permutations of stops in an effective way.

The column generation algorithm (step 2A) still performs well however. The average solution time of the 13 decomposed problems for problem TOR880 is 60 seconds. Each problem in the decomposition involves around 35 mini-clusters to be covered by over 40 different pieces of work i.e. over 40 different subproblems generating feasible routes. For the 5 problems solved by decomposition, the optimal integer solution was obtained for the small routing problems on mini-clusters 18 times out of 24. For the other 6 problems the gap with respect to the continuous simplex solution was 0.2% on average.

Finally, the total computation time of over 25 minutes for problem TOR880 may seem prohibitive. Recall that this problem is essentially an m-travelling salesman problem with two depots and 1760 nodes to be visited subject to time, capacity precedence, pairing and availability constraints. We believe that the quality of the solution obtained is encouraging for the use of mathematical optimization technique like those described in this article. Further work needs however to be done to improve file mangement time and to implement methodology more efficiently.

5. CONCLUSION

The problem of vehicle route construction for the transportation of the handicapped is difficult to solve, both because of the size of the problem and because of the multiplicity and complexity of the constraints. The construction of mini-clusters of clients travelling at the same time in the same vehicle lessens the difficulties by reducing the size of the new routing problem while satisfying in total or in part certain constraints (precedence, pairing, time windows). The new routing problem is however also difficult to solve and is subject to time window, availability and multiple depot constraints. But, through a mathematical programming methodology using column generation, these constraints can be handled in subproblems generating feasible routes.

Although the results are encouraging, more work needs to be done: the improvement of the single vehicle algorithm to handle stops which are very close together, the evaluation of different decomposition stategies, the implementation of more efficient code for the simplex, and better file management.

Research will also be carried out on the method of constructing mini-clusters and the impact of mini-clustering on solution quality. Along the same times, non-disjoint mini-clusters where several (i.e. mini-clusters might contain the same request) could be considered in the subproblems generating feasible routes. The covering equations of the master problem would then refer to the original requests rather than the mini-clusters.

REFERENCES

[1] J.-P. BÉLISLE, F. SOUMIS, S. ROY, J. DESROSIERS, Y. DUMAS and J.-M. ROUSSEAU, The Impact on Vehicle Routing of Various Operational Rules of a Transportation System for Handicapped Persons, Centre de recherche sur les transports, Université de Montréal, (1984).

[2] L. BODIN, B. GOLDEN, A. ASSAD and M. BALL, Routing and Scheduling of Vehicles and Crews - The State of the Art, Computer & Operations Research, Special Issue, (1983).

[3] L. BODIN and T. SEXTON, The Multi-vehicle Subscribe Dial-a-ride Problem. Management Science & Statistics Working Paper No. 82-005, University of Maryland at College Park, (1982).

[4] J. DESROSIERS, Y. DUMAS and F. SOUMIS, A Dynamic Programming Method for the Large Scale Single Vehicle Dial-a-ride Problem with Time Windows, Montréal, Centre de recherche sur les transports, No 361, Université de Montréal, (1984).

[5] J. DESROSIERS, P. PELLETIER and F. SOUMIS, Plus court chemin avec contraintes d'horaires, RAIRO 17(4), 357-377, (1983).

[6] J. DESROSIERS, F. SOUMIS and M. DESROCHERS, Routing with Time Windows by Column Generation, NETWORKS 14, (1984).

[7] J. JAW, A. ODONI, H. PSARAFTIS and N. WILSON, A Heuristic Algorithm for the Multi-vehicle Many to Many Advanced Request Dial-a-ride Problem, Version 2, Presented at the ORSA/TIMS Meeting, Houston, Texas, (1981).

[8] H. PSARAFTIS, An Exact Algorithm for the Single Vehicule Many-to-many Dial-a-ride Problem with Time windows, Technical Note OE-UMTA-82-2, MIT, Cambrige, Mass., (1982).

[9] H. PSARAFTIS, Analysis of an $O(N^2)$ Heuristic for the Single Vehicle Many-to-many Euclidian Dial-a-ride Problem, Transportation Research, 17B(2), 133-145, (1983).

[10] S. ROY, J.-M. ROUSSEAU, G. LAPALME and J.A. FERLAND, Routing and Scheduling for the Transportation of Disabled Persons- The Algorithm, Montréal, Transport Canada, Tranport Development Center, TP 5596 E, (1984).

[11] S. ROY, J.-M. ROUSSEAU, G. LAPALME and J.A. FERLAND, Routing and Scheduling for the Transportation fo Disabled Persons- The Tests, Montreal, Transport Canada, Transport Development Center, TP 5598 E (1984).

[12] T. SEXTON and L. BODIN, The Single Vehicle Many to Many Routing and Scheduling Problem with Desired Delivery Times, Working Paper No. No. 80-014, University of Maryland at College Park, (1980).

[13] M. SOLOMON, Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms, Working Paper No. 83-02-01, The Warton School, University of Pennsylvania, (1983).